



**FPT POLYTECHNIC**



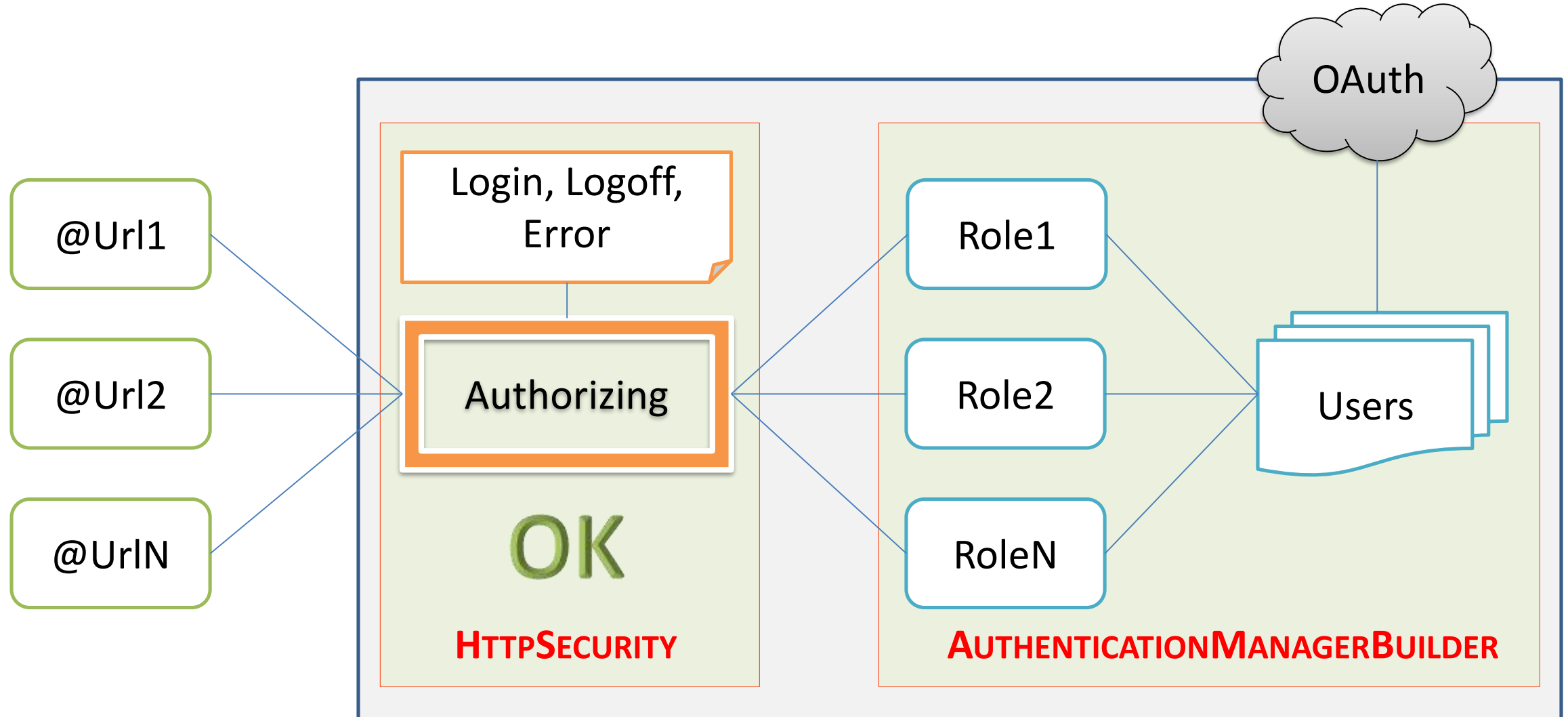
Conceive Design Implement Operate

## **ADVANCED SPRING SECURITY**

**GIẢNG VIÊN: NGUYỄN NGHIỆM**

[www.poly.edu.vn](http://www.poly.edu.vn)



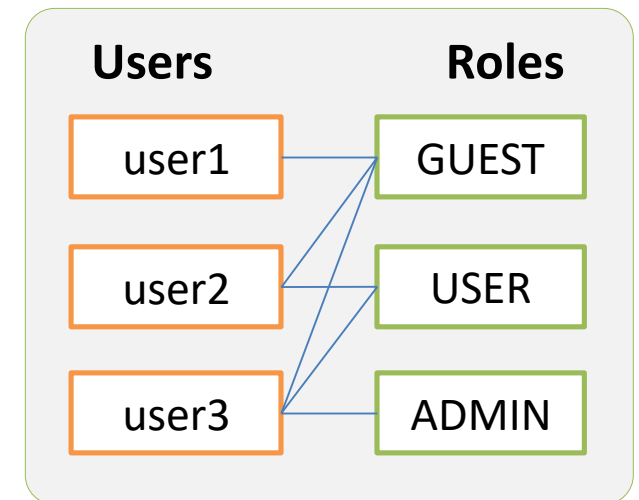


```
/*--QUẢN LÝ NGƯỜI DỮ LIỆU NGƯỜI SỬ DỤNG--*/
```

```
@Override
```

```
protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
    BCryptPasswordEncoder pe = new BCryptPasswordEncoder();  
    auth.inMemoryAuthentication()  
        .withUser("user1").password(pe.encode("123")).roles("GUEST")  
        .and()  
        .withUser("user2").password(pe.encode("123")).roles("USER", "GUEST")  
        .and()  
        .withUser("user3").password(pe.encode("123")).roles("USER", "GUEST", "ADMIN");  
}
```

- ❑ Nguồn dữ liệu cố định, không thực tế
- ❑ Cần tổ chức dưới dạng database, có thể phân quyền được thông qua ứng dụng





# AUTHENTICATION DATABASE

---

## Roles

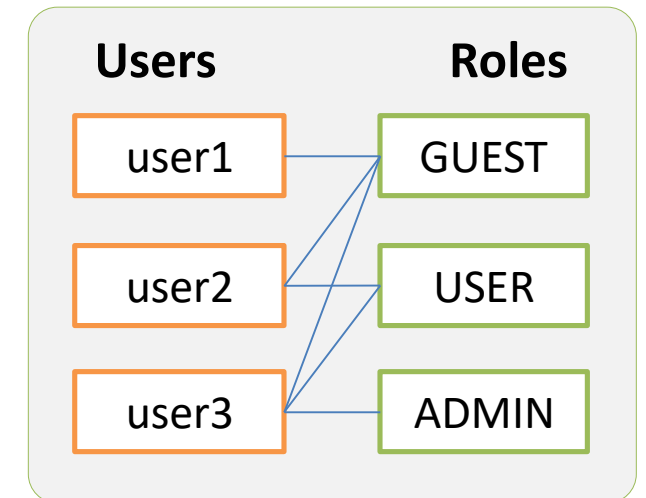
Id	Name
ADMIN	Administrators
GUEST	Guests
USER	Users

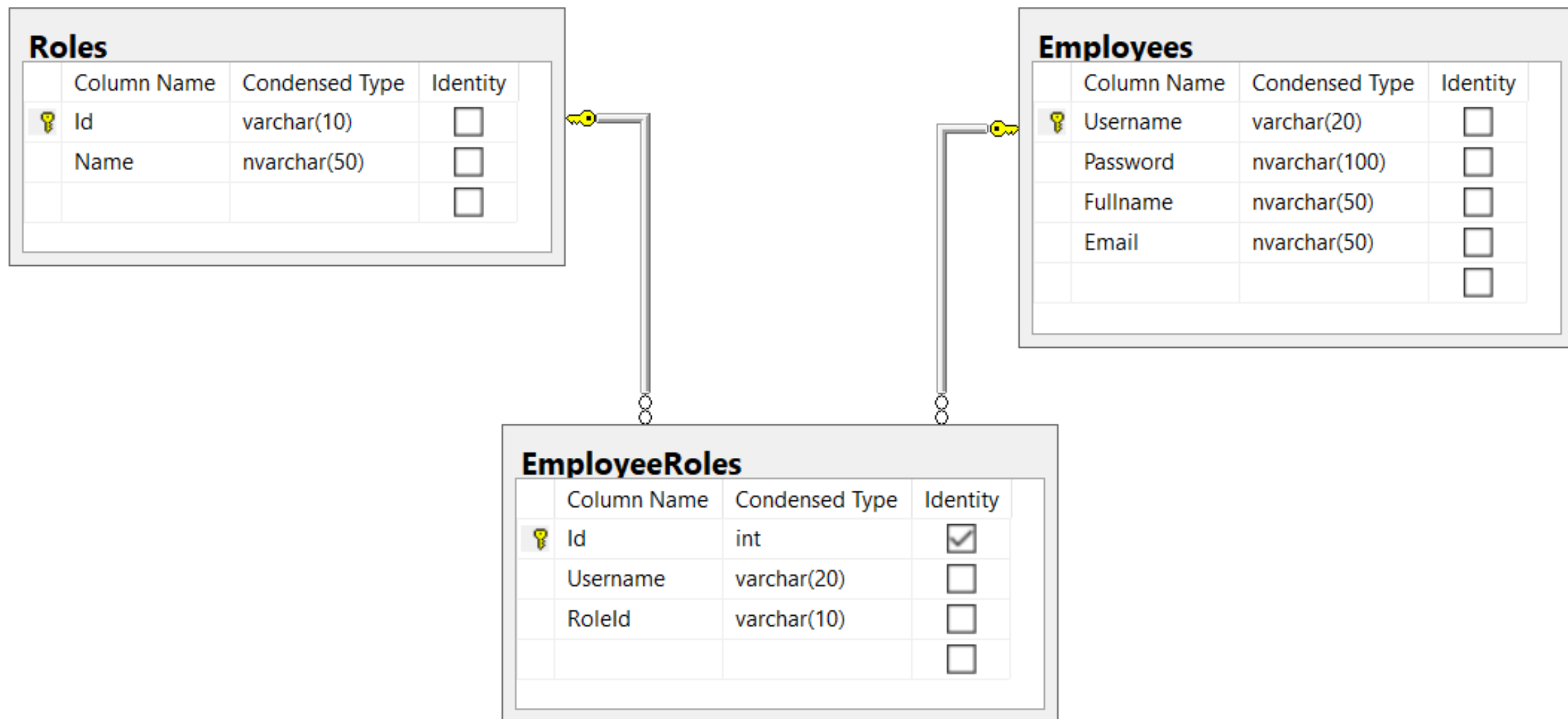
## Employees

Username	Password	Fullname	Email
user1	123	Nguyễn Văn User1	user1@fpt.edu.vn
user2	123	Đoàn Thị User2	user2@fpt.edu.vn
user3	123	Trần Thị Mỹ User3	user3@fpt.edu.vn
user4	123	Phạm Tuấn User4	user4@fpt.edu.vn

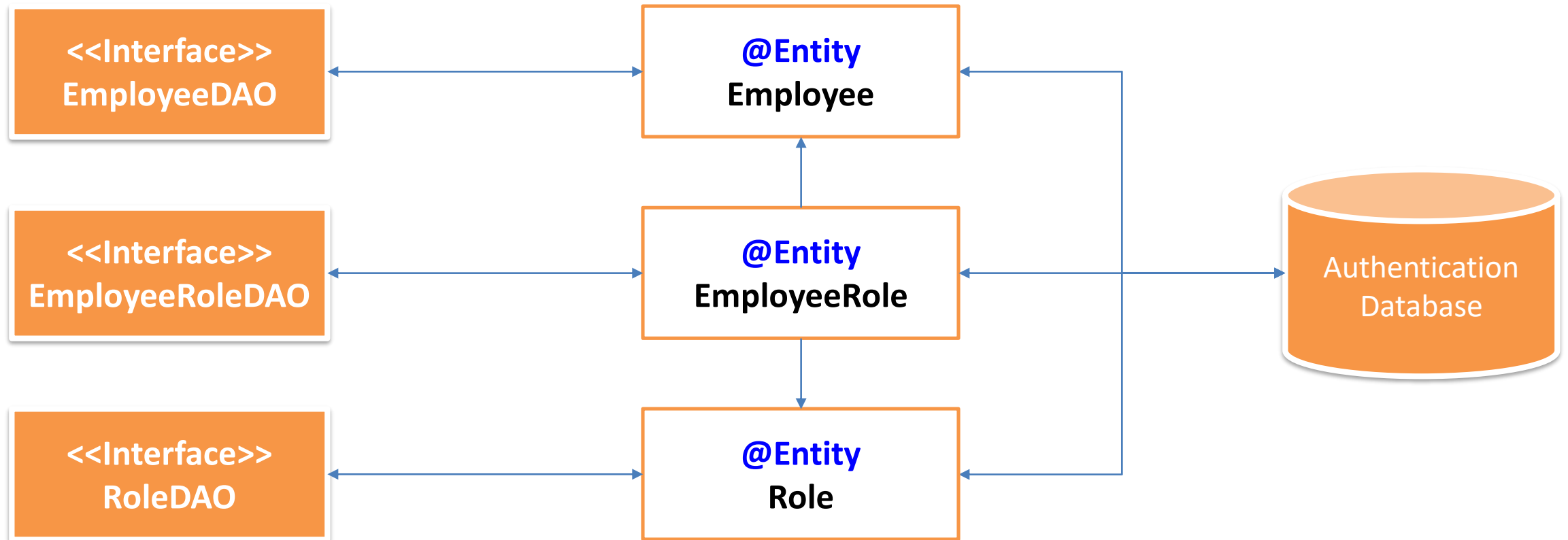
## EmployeeRoles

Id	Username	RoleId
1003	user2	USER
1004	user2	GUEST
1005	user3	ADMIN
1006	user3	USER
1007	user3	GUEST
2003	user1	GUEST









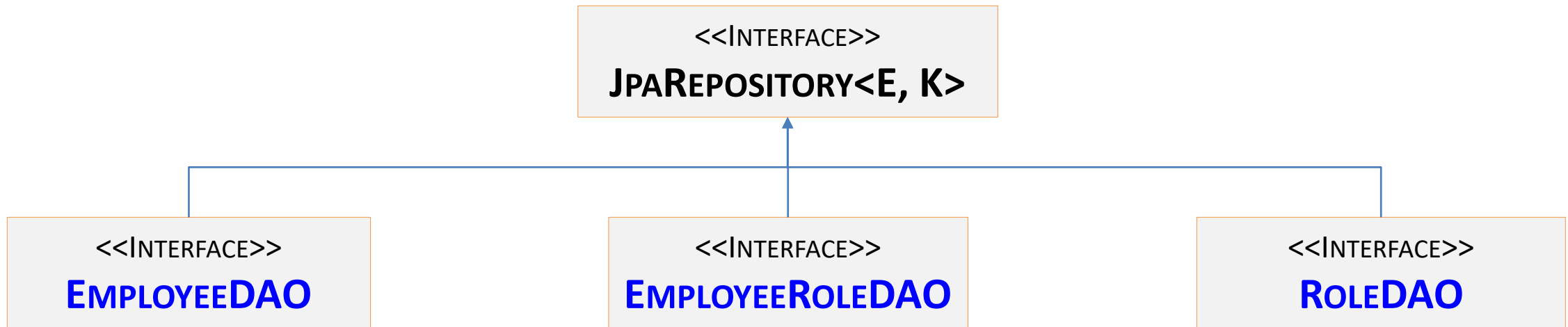


```
@Data
@Entity
@Table(name = "roles")
public class Role {
    @Id
    private String id;
    private String name;
    @JsonIgnore
    @OneToMany(mappedBy = "role")
    List<EmployeeRole> employeeRoles;
}
```

```
@Data
@Entity
@Table(name = "Employees")
public class Employee {
    @Id
    private String username;
    private String password;
    private String email;
    private String fullname;
    @JsonIgnore
    @OneToMany(mappedBy = "employee")
    List<EmployeeRole> employeeRoles;
}
```

```
@Data
@Entity
@Table(name = "EmployeeRoles")
public class EmployeeRole {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @ManyToOne @JoinColumn(name = "Username")
    private Employee employee;
    @ManyToOne @JoinColumn(name = "Roleid")
    private Role role;
}
```

```
graph TD
    ER[EmployeeRole] --> R[Role]
    ER --> E[Employee]
```



```
public interface EmployeeDAO extends JpaRepository<Employee, String> {
}
public interface RoleDAO extends JpaRepository<Role, String> {
}
public interface EmployeeRoleDAO extends JpaRepository<EmployeeRole, Integer> {
}
```

pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>mssql-jdbc</artifactId>
  <scope>runtime</scope>
</dependency>
```

application.properties

```
spring.datasource.url=jdbc:sqlserver://localhost;database=PolyJ6
spring.datasource.username=sa
spring.datasource.password=*****
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto = none
```

Polytechnic

x

+



localhost:8080/authorize/index



## AUTHORIZATION

	ADMIN	GUEST	USER
user1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
user2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
user3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
user4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

{  
[

}  
]

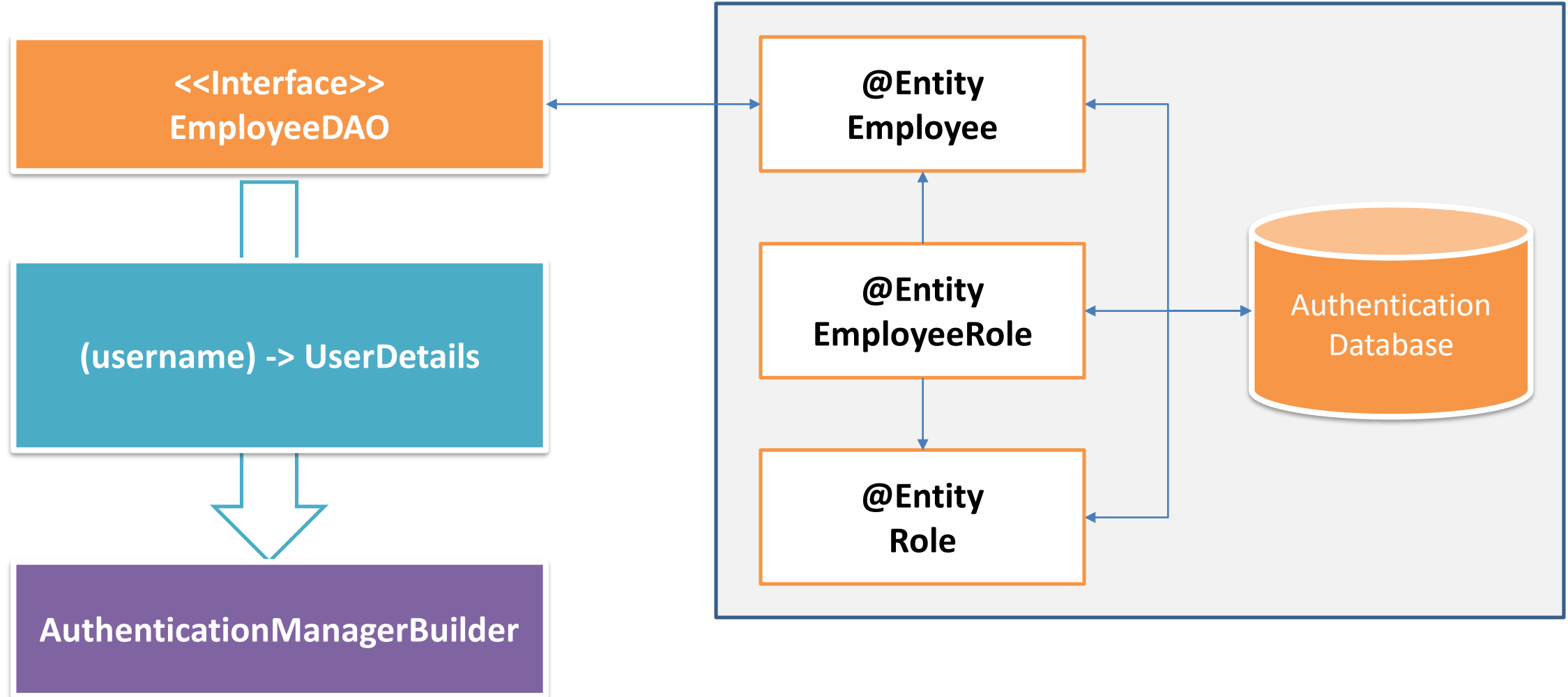
**DEMO**





# AUTHENTICATION MANAGER

---





# LOAD USER FROM DATABASE IMPLEMENTATION

```
@Autowired
EmployeeDAO edao;

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(username -> {
        try {
            Employee employee = edao.findById(username).get();
            String password = employee.getPassword();
            String[] roles = employee.getEmployeeRoles().stream()
                .map(er -> er.getRole().getId())
                .collect(Collectors.toList()).toArray(new String[0]);

            return User.withUsername(username).password(password).roles(roles).build();
        } catch (Exception e) {
            throw new UsernameNotFoundException(username + " not found!");
        }
    });
}
```

{  
[

}  
]

**DEMO**





# OAuth2 AUTHENTICATION

---

Username?

Password?

☐ Remember me?

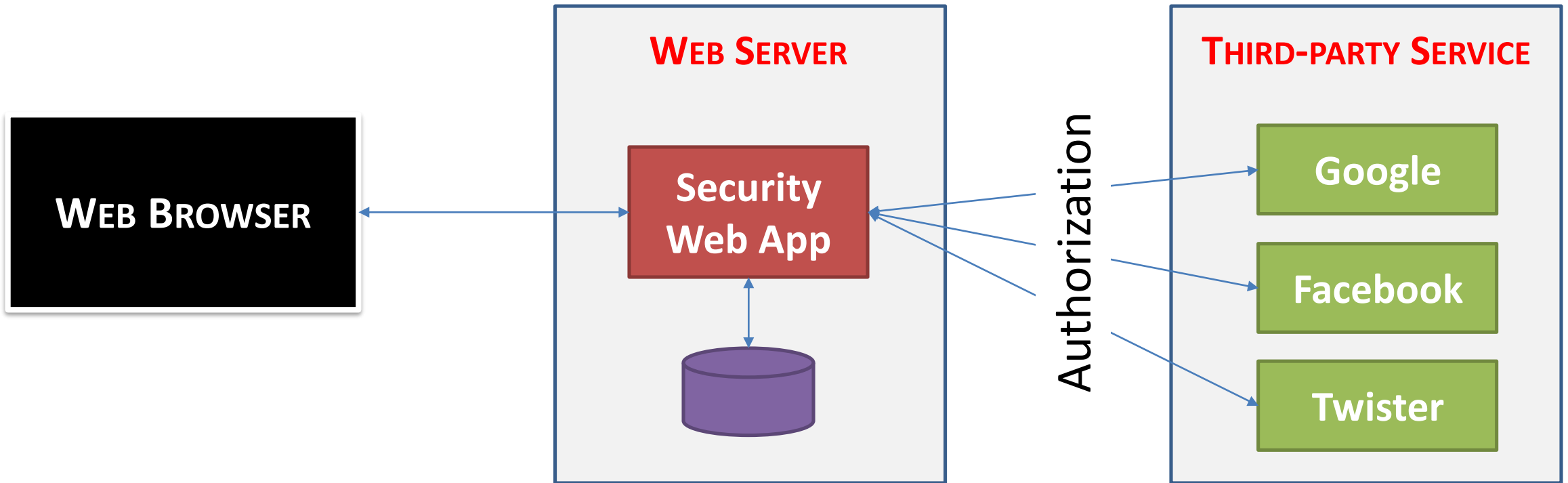
Login

Google

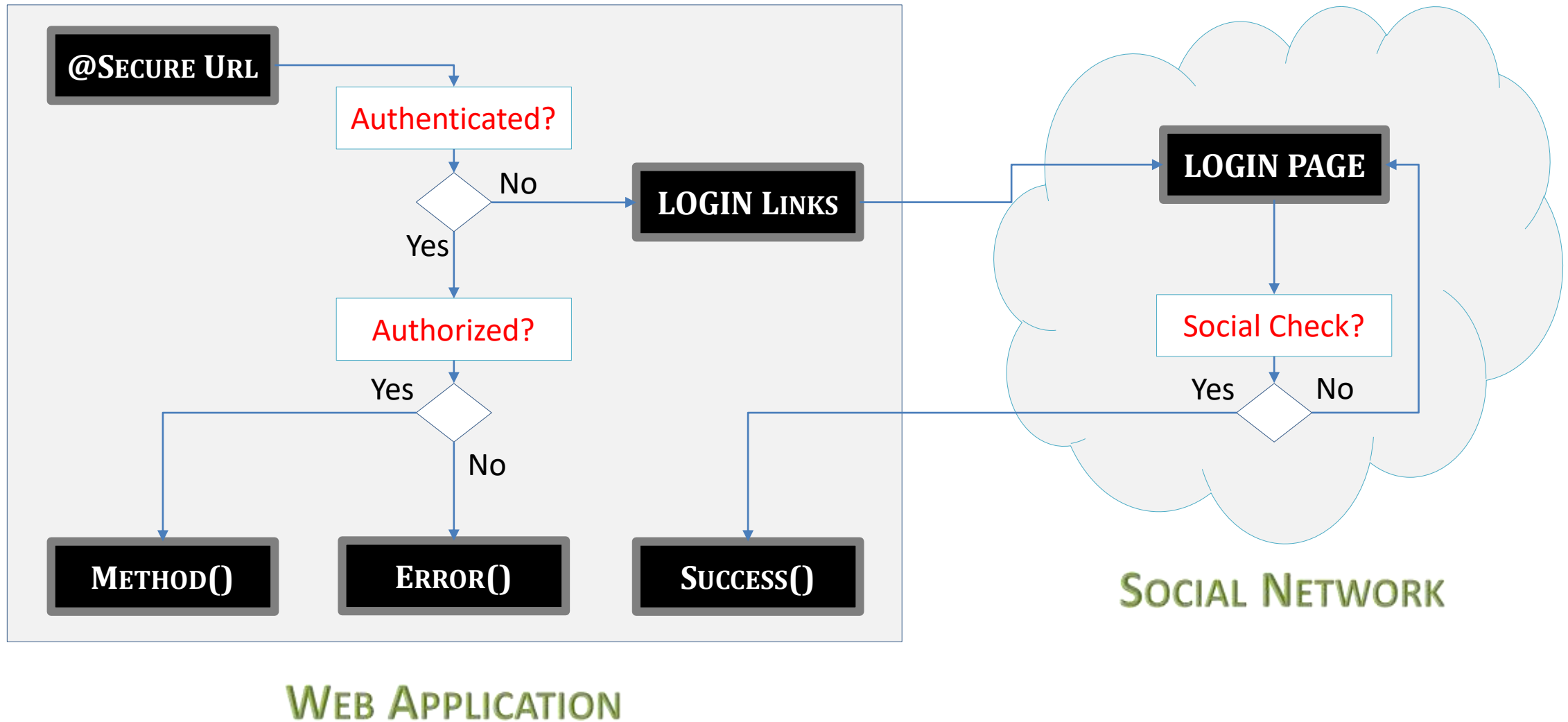
facebook



- ❑ **OAuth** (Open Authorization) là một framework ủy quyền chuẩn mở dựa trên nền tảng internet.
- ❑ **OAuth** (phát âm là "oh-auth") cho phép các dịch vụ của bên thứ ba như Facebook và Google..., cung cấp thông tin tài khoản của người sử dụng mà không để lộ.



*Cần phải đăng ký với mạng xã hội để lấy khóa bí mật (**app-id** và **secret**) khai báo vào Web Application để xác thực thông tin người dùng trong quá trình hoạt động*





- ❑ Tạo tài khoản developer trên Google và Facebook để lấy **credentials** (*app-id* và *secret code*)
  - ❖ Google API Console
    - <https://console.developers.google.com/>
  - ❖ Facebook for Developers
    - <https://developers.facebook.com/docs/facebook-login>
- ❑ Khai báo Redirect URI (sau khi đăng nhập thành công)
  - ❖ Google API Console (Authorized Redirect URI)
    - <http://localhost:8080/login/oauth2/code/google>
  - ❖ Facebook for Developers (Valid OAuth Redirect URI)
    - <http://localhost:8080/login/oauth2/code/facebook>
- ❑ Lấy credentials (*app-id* và *secret*) để khai vào application.properties

## ❑ Google

- ❖ spring.security.oauth2.client.registration.google.client-id= <app-id>
- ❖ spring.security.oauth2.client.registration.google.client-secret= <secret>

## ❑ Facebook

- ❖ spring.security.oauth2.client.registration.facebook.client-id = <app-id>
- ❖ spring.security.oauth2.client.registration.facebook.client-secret= <secret>

## ❑ Thư viện cần thiết

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-oauth2-client</artifactId>  
</dependency>
```

# OAuth2 CONFIGURATION

```

http.oauth2Login()
    .loginPage("/oauth2/login/form")
    .defaultSuccessUrl("/oauth2/login/success", true)
    .failureUrl("/oauth2/login/error")
    .authorizationEndpoint() // Authorization Request
        .baseUrl("/oauth2/authorization")
        .authorizationRequestRepository(getRepository())
    .and().tokenEndpoint() // Authorization Response
        .accessTokenResponseClient(getToken());

```

```

<a href="/oauth2/authorization/google">
<a href="/oauth2/authorization/facebook">

```

```

@Bean
public AuthorizationRequestRepository<OAuth2AuthorizationRequest> getRepository() {
    return new HttpSessionOAuth2AuthorizationRequestRepository();
}


```

```

@Bean
public OAuth2AccessTokenResponseClient<OAuth2AuthorizationCodeGrantRequest> getToken() {
    return new DefaultAuthorizationCodeTokenResponseClient();
}

```

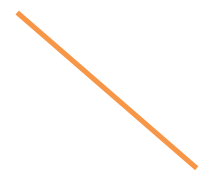
```
//.loginPage("/oauth2/login/form")  
@GetMapping("/oauth2/login/form")  
public String form() {  
    return "oauth2/login";  
}
```



```
<!--.baseUri("/oauth2/authorization")-->  
<a href="/oauth2/authorization/google">Google</a>  
<a href="/oauth2/authorization/facebook">Facebook</a>
```

```
//.defaultSuccessUrl("/oauth2/login/success")  
@GetMapping("/oauth2/login/success")  
public String success(OAuth2AuthenticationToken oauth) {...}
```

```
//.failureUrl("/oauth2/login/error")  
@GetMapping("/oauth2/login/error")  
public String error() {...}
```



Lấy thông tin từ mạng xã hội ở đây để thực hiện đăng nhập

```
@GetMapping("/oauth2/login/success")
public String success(OAuth2AuthenticationToken oauth) {

    // ĐỌC THÔNG TIN TÀI KHOẢN TỪ MẠNG XÃ HỘI
    String email = oauth.getPrincipal().getAttribute("email");
    String fullname = oauth.getPrincipal().getAttribute("name");

    // TẠO ĐỐI TƯỢNG USERDETAILS (PRINCIPAL) LẤY EMAIL LÀM USERNAME
    UserDetails user = User.withUsername(email).password("").roles("GUEST").build();

    // TẠO ĐỐI TƯỢNG AUTHENTICATION TỪ USERDETAILS
    Authentication auth =
        new UsernamePasswordAuthenticationToken(user, null, user.getAuthorities());

    // THAY ĐỔI THÔNG TIN ĐĂNG NHẬP CỦA HỆ THỐNG
    SecurityContextHolder.getContext().setAuthentication(auth);

    return "oauth2/success";
}
```

{  
[

}  
]

**DEMO**







# CONSUME SECURED REST API

---



- ❑ Để truy xuất các REST API bảo mật thì request phải chứa header Authorization có định dạng:

**“Authorization”: “Basic <auth>”**

❖ <auth> là chuỗi mã hóa dạng **base 64** của **“username:password”**

- ❑ Với JavaScript:

❖ **btoa**(“username:password”);

- ❑ Với Java (java.util.Base64):

❖ **Base64.getEncoder().encodeToString**(“username:password”.getBytes())

```
var app = angular.module("app", []);  
app.config(function($httpProvider) {  
    var auth = `Basic ${btoa("user2:123")}`;  
    $httpProvider.defaults.headers.common["Authorization"] = auth;  
})  
app.controller("ctrl", function($http){  
    $http.get("http://localhost:2021/auth/rest")  
        .then(response => console.log(response))  
        .catch(error => console.log(error))  
})
```

## CONSUME SECURED REST API USING URL

```
URL url = new URL("http://localhost:2021/auth/rest");  
URLConnection conn = (URLConnection) url.openConnection();  
conn.setRequestProperty("Accept", "application/json");  
conn.setRequestMethod("GET");  
String auth = Base64.getEncoder().encodeToString("user2:123".getBytes());  
conn.setRequestProperty("Authorization", "Basic " + auth);
```

Sử dụng **exchange()** thay vì **getForObject()**, **postForObject()**, **put()** và **delete()**

```
RestTemplate client = new RestTemplate();
```

```
//1. Headers
```

```
HttpHeaders headers = new HttpHeaders();
```

```
String auth = Base64.getEncoder().encodeToString("user2:123".getBytes());
```

```
headers.add("Authorization", "Basic " + auth);
```

```
//2. Data (NULL nếu GET hoặc DELETE)
```

```
HttpEntity<T> request = new HttpEntity<>(data, headers);
```

```
//3. Tạo Request & nhận Response
```

```
String url = "http://localhost:2021/auth/rest";
```

```
ResponseEntity<R> response =
```

```
    client.exchange(url, HttpMethod.GET, request, R.class);
```

GET, POST, PUT, DELETE

{  
[

}  
]

**DEMO**





- ✓ Authentication Database
  - ✓ Entities và DAO
- ✓ Authentication Manager
  - ✓ (username) -> UserDetails
  - ✓ User: hỗ trợ tạo UserDetails
- ✓ OAuth2
  - ✓ Developer account, lấy app-id và secret
  - ✓ Khai báo pom.xml và application.properties
  - ✓ Cấu hình `HttpSecurity.oauth2Page()`
  - ✓ Xử lý thông tin đăng nhập lấy từ mạng xã hội
- ✓ Consume Secured REST API
  - ✓ Authorization: Basic <auth>





**Cảm ơn**